

Superscalar Performance in a Multithreaded Microprocessor

by

Bernard Karl Gunther, BE (Hons)

Department of Computer Science

Submitted in fulfilment of the requirements
for the degree of
Doctor of Philosophy

UNIVERSITY OF TASMANIA
HOBART

December 1993

Statement of Originality

This thesis does not contain any material which has been accepted for the award of any other higher degree or graduate diploma in any tertiary institution. To the best of my knowledge and belief, this thesis does not contain any copy or paraphrase of material previously published or written by another person, except when due reference is made in the text of the thesis.

B. Gonth

Access and Copying

This thesis may be made available for loan and limited copying in accordance with the Australian Copyright Act 1968.

B. Smith

19/10/44

Abstract

Multithreaded processors, having hardware support for the concurrent execution of fine-grained threaded computations, are noted for their latency tolerance and low-cost synchronization. Multithreading is a technique for improving the utilization of processing elements (PEs) in parallel processing systems, thereby reducing cost/performance ratios. With increasing integrated circuit densities it is becoming feasible to integrate several PEs onto a single die, and further diminish the physical dimensions of parallel systems. However, by eliminating the artificial on-chip PE boundaries and sharing expensive resources in a more tightly coupled multithreaded architecture, even greater performance can be achieved from similar hardware.

A multithreaded processor architecture (Concurro) was designed for possible microprocessor implementation with the objective of multiple instruction issues per cycle—sustained superscalar performance—by means of multithreading. This thesis considers the trade-offs necessary for such architectures to achieve high throughput and hardware utilization under scalability and cost constraints. A detailed simulation study was carried out to characterize the architecture and evaluate the impact of implementation decisions. The key to efficiency in Concurro is asynchronous, zero-time context switching among a limited set of contexts, promoting effective use of the storage hierarchy. A 64-bit, register-based, load/store instruction set architecture is augmented with thread manipulation primitives and I-structure synchronization operations. Novel cache architectures and controller algorithms were designed for enhancing latency tolerance in the processor, while maximizing utilization of the most costly resources.

When tested on a variety of numerical and integer workloads, Concurro was able to sustain superscalar instruction issue rates for multithreaded operation, yet showed scalar RISC performance on single-thread code. Even with a simple threading strategy it was frequently possible to extract full utilization from functional units or the instruction cache. The architecture showed size scalability to an order of magnitude while remaining binary

compatible across these configurations. Performance of large configurations was shown to be limited ultimately by the bandwidth available from critical shared resources. With an appropriate memory system Concurro attained supercomputer-level floating point throughput operating out of uncached memory. The hardware requirements for this performance are expected to be comparable with those of VLIW machines with similar datapaths.

Acknowledgments

I thank my supervisor, Arthur Sale, for his continued confidence in me, despite my occasional meanderings, and for teaching me the value of independent and critical enquiry. I am indebted to John Morris for reading my drafts with care and enthusiasm, and returning them with valuable feedback. Andrew Partridge's reviews and insights are also very much appreciated.

I am grateful to Carl Lewis for the initial development of the assembler, which helped in getting the first simulations started. Thanks to John Parry for his efforts with the lcc compiler back-end.

The support of many colleagues from the old GCSB will be long remembered. I thank Ed Kazmierczak, Simon Milton, David Wright, Andrew Partridge, Jo Jordan, Tony Dekker, and Ben Lian for their companionship and advice, and for sharing with me their spirit of discovery. In particular, I wish to thank David Wright for introducing to me some fascinating perspectives of computer science.

I am especially grateful to my mother for her selfless help, patience, and interest. I thank her for instilling in me the courage to pursue my goals.

B.K. Gunther
Hobart, 1993

*To my mother,
Eva.*

Contents

Statement of Originality	ii
Access and Copying	iii
Abstract	iv
Acknowledgments	vi
1 Introduction	1
1.1 Multithreading	2
1.2 Related Work	11
1.3 This Thesis	24
2 Core Architecture	26
2.1 Instruction Set Architecture	27
2.2 System Architecture	36
2.3 Processor Organization	38
2.4 Instruction Pipeline	48
3 Subsystem Architecture	52
3.1 Context Units	53
3.2 Instruction Cache	59
3.3 Functional Units	63
3.4 Data Cache	67
4 Experimental Method	75
4.1 Simulation Technique	76
4.2 Benchmark Programs	79
4.3 Performance Evaluation	86
5 Organization Alternatives	88
5.1 Scalability	89
5.2 Instruction Cache Duplication	96
5.3 Multiprocessing	99
5.4 Implementation Cost	103
5.5 Observations and Conclusions	105

6	Instruction Fetch and Dispatch Strategies	107
6.1	Instruction Fetching and Scheduling	108
6.2	Branching.....	117
6.3	Instruction Cache Performance	122
6.4	Observations and Conclusions.....	127
7	Execution Resources	128
7.1	Utilization	129
7.2	Latency Tolerance	135
7.3	Datapath Bandwidth.....	141
7.4	Data Cache Performance	148
7.5	Observations and Conclusions.....	155
8	Conclusion	157
8.1	Major Microarchitectural Features	158
8.2	Reduced-Cost Designs	160
8.3	Future Work	162
A	Concurro Instruction Set	164
B	Synchronization Controller Microprogram	172
C	Processor Configurations	176
	Bibliography	180